UNITED STATES PATENT AND TRADEMARK OFFICE

————————

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

————————

Ex parte EITAN FARCHI, THOMAS JOSEPH PAVELA,
SHMUEL UR, and AVI ZIV

————————

Appeal 2006-2958
Application 09/990,802
Technology Center 2100

————————

Decided: November 30, 2007

————————

Before LANCE LEONARD BARRY, HOWARD B. BLANKENSHIP,
ALLEN R. MACDONALD, JEAN R. HOMERE, and JAY P. LUCAS,
Administrative Patent Judges.

HOMERE, *Administrative Patent Judge*.

DECISION ON REQUEST FOR REHEARING

STATEMENT OF THE CASE

In a paper filed June 20, 2007, Appellants request reconsideration
under 37 C.F.R. § 41.52 from a Decision of the Board of Patent Appeals and
Interferences dated February 16, 2007 (Decision, hereinafter.)

In the Decision, the Board affirmed the Examiner's rejection of claims 1 through 21.

Appellants contend that the Decision does not give sufficient guidance as to why the many arguments they advanced were inadequate to overcome the Examiner's rejections. (Req. Reh'g 1.) Further, Appellants allege that the Board Decision does not appear to have addressed all of the arguments that Appellants advanced in the Appeal Brief and Reply Brief. (*Id*. 2.)

We grant the Request.

ISSUES

First, Appellants argue that "While the Board said the Examiner found properly that the code persisted across code versions, the Board did not address Appellants' detailed argument in this regard, and in particular did not appear to address specifically Appellants' arguments that 'across' means 'on both sides of,' while Chen clearly teaches something less than that." (Req. Reh'g 2-3.) Therefore, the first issue is whether Appellants have shown that the Decision failed to address the argument that Chen's codes do not persist *across* (meaning on both sides of) software versions, as submitted in the Briefs?

Second, Appellants argue that:

> "While the Board chose to focus only on the aspect of the invention related to the persistence of code coverage tasks, Appellants' discussion of code coverage tasks took into account the dividing, inserting, and creating steps in independent claim 1 and

independent claim 8 and independent claim 15. The Board's Decision addressed none of this. Again, the Board's Decision focused only on the aspect of persistence." (Req. Reh'g 4-5.)

Particularly, Appellants contend that the Decision merely agrees with the Examiner that Chen's "basic code entities" are equivalent to the claimed "code coverage tasks". However, the Decision does not provide sufficient details explaining why the two codes are the same. (*Id.* 4.) Therefore, the second issue is whether Appellants have shown that the Decision failed to provide sufficient reasoning to support the conclusion that Chen's basic code entities reasonably teach Appellants' code coverage tasks?

Third, Appellants contend that the Decision does not appear to address the argument that Winder has nothing to do with code coverage tasks or basics entities, and therefore would not be helpful in modifying Chen. (*Id.* 5.) Therefore, the third issue is whether Appellants have shown that the Decision failed to address the argument that Winder is not analogous art, and therefore not combinable with Chen?

FINDINGS OF FACT

The following findings of fact are supported by a preponderance of the evidence.

The Invention

1. Appellants invented a method, apparatus and article of manufacture for selective regression testing of a software suite. (Spec. 1.)

2. Particularly, the invention divides a program source code into a plurality of basic blocks of codes (code coverage tasks) and assigns a persistent unique name to each of the plurality code coverage tasks.[1] (*Id*. 12.) The created persistent unique names are then collected across software versions to identify which subsets of the test suite must be run in order to test a new version of the software. (*Id*. 13.)

Prior Art Relied Upon

3. Chen discloses a regression testing method and system for selecting which test units of an existing software must be re-run in order to test a new version of the software. (Col. 1, ll. 10-12.)

4. As depicted in Figure 1, Chen teaches partitioning the source code of the existing software version (30) into basic code entities,[2] grouped as a plurality of test units (46, 48, 50) in a test suite (51). (Col. 2, ll. 47-49.)

---

[1] Appellants' Specification, at pages 12-13, defines code coverage task as a "basic block of code for which an execution of a test returns a true value if the testing requirement of the task is fulfilled and a false value if the testing requirement of the task is not fulfilled." "[A] basic block is a set of consecutive statements with a single entry point (i.e. the first statement) and a single exit point (i.e. the last statement)."

[2] Chen describes each basic code entity as being a function (F) or a non-function (V) linked together as part of a test unit. (Col. 4, ll. 54-62.) An entity is considered to be changed if the source code text of the definition of the entity has changed. (Col. 11, ll. 1-3.) Further, Chen discloses an entity trace list for each test unit that tracks the entities covered by each test unit. (Col. 9, ll. 31-35.)

4

5. As depicted in Figure 4, Chen discloses the creation of a trace list (161, 163, 165) for each test unit (160, 162, 164) to keep track of the entities that are covered by each test unit. (Col. 9, ll. 32-35.)

6. Chen further discloses that each entity has, inter alia, an entity kind, a name and a checksum. (Col. 9, ll. 1-2). The checksum is used to determine whether an entity has been changed after a revision to the software system. (Col. 8, ll. 59-62.)

7. Chen discloses an unchanged entity list (246) for entities which are unchanged between a first source code program and a second source code program. (Col. 11, ll. 9-12.)

8. Chen also discloses that two entities match if they have the same name and entity kind. And, if the checksums of the two entities match, the source code texts of the two entities are probably identical. (Col. 11, ll. 20-21, ll. 46-49).

9. Entity attributes are compared across software versions to determine which test units to re-run depending on which entities were previously modified. (Col. 2, ll. 47-60; col. 12, ll. 9-18.)

10. Winder discloses using naming conventions in software management. Particularly, Winder teaches selecting meaningful program variables and routine names in software development so they can be readily identified and reused. (Page 84.)

ANALYSIS

Issue I

We have carefully reviewed our original Decision in light of Appellants' remarks. We find that we did address Appellants' argument pertaining to the claimed limitation "*across*". Further, we find no error in the analysis or logic set forth in our original Decision regarding the discussion of such limitation.[3] Particularly, the Decision explained in detail our rationale for agreeing with the Examiner that Chen's basic code entities do persist *across* software versions. (See Decision 9, ll. 1-8.) As noted in the Decision, we found that Chen discloses, in a current software version, testing is performed only for test units of a prior software version having at least one modified entity. (*Id.*) We also found that test units with no modified entities in the prior software version are reused in the current software version. (*Id.*) Consequently, the test units with no modified entities are reused in both the prior software version and the current software version. (*Id.*) Thus, we concluded that unmodified entities in test units persist *across* the two software versions. (*Id.*) For the foregoing reasons,

---

[3] As detailed in the Findings of Fact section, herein, we have found (once again) that Chen discloses that test units with no modified entities in a prior software version are re-used, and they need not be tested in a current software version. (Findings 7 through 9.) Therefore, it is reasonable to conclude that unmodified entities in such test units persist *across* (on both sides, i.e. used in both prior and current versions) of the two software versions.

we decline to modify our discussion of the claim limitation "*across*" in the Decision.

Issue 2

In response to Appellants' request for clarification, we have modified the prior Decision as follows:

We agree with the Examiner that Chen's disclosure of the basic code entities reasonably teaches the claimed "code coverage tasks." As set forth in the Findings of Fact section above, we have found that Chen discloses partitioning the source code of a software version into basic code entities linked together as functions and non-functions. (Finding 4.) Further, we have found that Chen discloses creating an entity trace list for each test unit thereby delimiting the source code by grouping entities covering each respective test unit. (*Id*.) Additionally, we have found that entities of different software versions are compared to one another to determine whether they have been modified. (Findings 6, 7.) A determination to place an existing entity on the unchanged entity list in a current software version depends on whether or not the entity has been modified. In other words, entities in a prior software version that remain unchanged in the current version are placed in the unchanged entity list. (*Id*.) It follows that to persist *across* software versions, Chen's disclosed entities must return a true value when the comparison indicates that they have not been modified.

Conversely, they must return a false value when it is determined that they have been modified.

Therefore, we find no error in the Examiner's finding that Chen's basic code entities teach the claimed code coverage tasks, as defined by Appellants.[4]

Issue 3

In response to Appellants' request for clarification, we have modified the prior Decision as follows:

Appellants' arguments are not persuasive. As set forth in the Findings of Fact section above, we have found that Chen is concerned with determining which entities in a prior software version are reusable in a current software version. (Finding 4.) We have also found that Winder[5] is particularly concerned with selecting meaningful program and variable names so they can be reused in a subsequent program. (Finding 10.) We find therefore that one of ordinary skill in the art would have readily recognized that Winder's teachings would provide Chen's entities with meaningful names, which would *predictably*[6] result in reinforcing the

_____

[4] See supra note 1.
[5] We note that the Examiner relied upon Winder for its teaching of using naming conventions in software management. (Ans. 13.)
[6] The Supreme Court has held that in analyzing the obviousness of combining elements, a court need not find specific teachings, but rather may consider "the background knowledge possessed by a person having ordinary skill in the art" and "the inferences and creative steps that a person of

reusability of such entities in subsequent software versions. Consequently, we find no error in the Examiner's conclusion that Chen and Winder can be properly combined to render the claimed invention unpatentable.

DECISION

In light of the foregoing, we grant Appellants' Request for Rehearing to the extent that we modify our prior Decision to further clarify our rationale for affirming the Examiner's rejections.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv).

---

ordinary skill in the art would employ." *KSR Int'l Co. v. Teleflex Inc.*, 127 S. Ct. 1727, 1740-41 (2007). To be nonobvious, an improvement must be "more than the predictable use of prior art elements according to their established functions." *Id.* at 1740.

REQUEST FOR REHEARING GRANTED

rwk

SUGHRUE MION PLLC
USPTO CUSTOMER NO WITH IBM/SVL
2100 PENNSYLVANIA AVENUE, N.W.
WASHINGTON DC 20037